

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

TRANSMITTAL OF APPEAL BRIEF (Large Entity)

BOARD OF PATENT
APPEALS &
INTERFERENCES

Docket No.
PO999046US1

In Re Application Of: D. F. Ault et al

JUL -1 2004

Application No.	Filing Date	Examiner	Customer No.	Group Art Unit	Confirmation No.
09/404,903	September 24, 1999	V. H. Nguyen		2126	8888

Invention: Method and Apparatus For Serializing A Message Queue In A Multiprocessing Environment

RECEIVED

JUL 06 2004

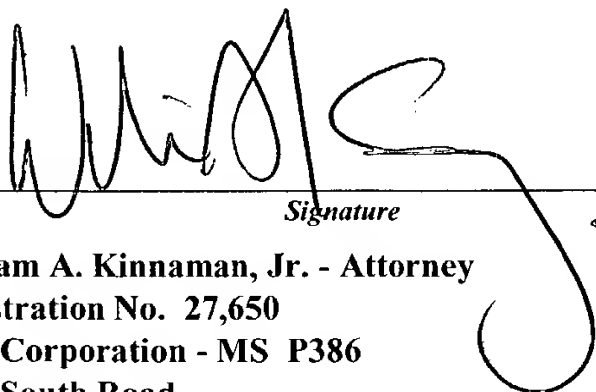
COMMISSIONER FOR PATENTS:

Technology Center 2100

Transmitted herewith in triplicate is the Appeal Brief in this application, with respect to the Notice of Appeal filed on May 4, 2004

The fee for filing this Appeal Brief is: \$330.00

- ☐ A check in the amount of the fee is enclosed.
- ☒ The Director has already been authorized to charge fees in this application to a Deposit Account.
- ☒ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. 09-0463



Signature

William A. Kinnaman, Jr. - Attorney
Registration No. 27,650
IBM Corporation - MS P386
2455 South Road
Poughkeepsie, NY 12601

Dated: June 29, 2004

I certify that this document and fee is being deposited on 06/29/2004 with the U.S. Postal Service as first class mail under 37 C.F.R. 1.8 and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.



Signature of Person Mailing Correspondence

Sandra L. Kilmer

Typed or Printed Name of Person Mailing Correspondence

CC:

Patent

IN THE U.S. PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Applicant: DONALD F. AULT et al.	: Group Art Unit: 2126	RECEIVED JUL 06 2004 Technology Center 2100
Serial No.: 09/404,903	: Examiner: Van H. Nguyen	
Filed: September 24, 1999	: June 29, 2004	
Confirmation No.: 8888	: William A. Kinnaman, Jr.	
Title: METHOD AND APPARATUS FOR SERIALIZING A MESSAGE QUEUE IN A MULTIPROCESSING ENVIRONMENT	: International Business Machines Corporation 2455 South Road, Mail Station P386 Poughkeepsie, NY 12601	

APPLICANTS' APPEAL BRIEF

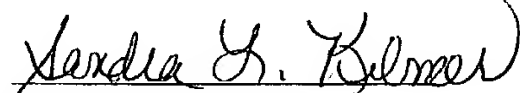
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Applicants hereby submit their appeal brief in the above-identified application.

CERTIFICATE OF MAILING UNDER 37 CFR 1.8(a)

I hereby certify that this correspondence is being deposited with the United States Postal Service as first-class mail in an envelope addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on June 29, 2004.


Sandra L. Kilmer

June 29, 2004
Date:

REAL PARTY IN INTEREST

The real party in interest is International Business Machines Corporation, the assignee of record.

RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

STATUS OF CLAIMS

Claims 1-43 are currently pending. Of these, claims 1-10, 24-26 and 34-36 stand rejected and are on appeal, while claims 11-23, 27-33 and 37-43 stand withdrawn from consideration. No claims have been allowed, nor have any been cancelled.

STATUS OF AMENDMENTS

There are no amendments after final rejection, unentered or otherwise.

SUMMARY OF INVENTION

The invention relates to a method, apparatus and program storage device for manipulating a queue and, more particularly, to a method, apparatus and program storage device for performing a recoverable operation on a message queue—represented in applicants' embodiment by a queue block (QB) 200 (Fig. 2)—in response to a request by a caller in an information handling system (Fig. 11: 10).

In accordance with the invention, there is stored a use count (Fig. 1: USE-COUNT 116) for the message queue indicating a count of tasks accessing the queue, as well as a use count flag (Fig. 3: USECNTFLAG 328) for the caller indicating whether the caller has acquired a lock on the queue. Atomically with updating the use count, the use count flag is updated to indicate whether

the caller has acquired a lock on the queue (page 16, lines 8-9). Applicants' invention allows one to serialize a message queue in a multiprocessing environment without the use of a conventional lock or latch to control access to the message queue data structures.

If the recoverable operation is a locking operation, the use count may be updated by incrementing it, while the use count flag may be updated to indicate that the caller has acquired a lock on the queue. Similarly, if the recoverable operation is an unlocking operation, the use count may be updated by decrementing it, while the use count flag may be updated to indicate that the caller has released a lock on the queue.

Preferably, the use count is compared with a previously read use count atomically with the updating steps, and the updating steps are performed only if the use count matches the previously read use count.

The use count may be stored in a message queue table (Fig. 1: 100) having an entry for the message queue. The message queue table may store an identifier (114) of the message queue, as well as a pointer (118) to the queue that is compared with a previously read pointer atomically with the updating steps; the updating steps are then performed only if the pointer matches the previously read pointer.

The use count flag may be stored in a control block (Fig. 3: 300) for the caller, which may also contain an identifier (301) of the message queue.

Preferably the updating steps are performed by executing a single atomic instruction—such as the Perform Lock Operation (PLO) instruction described—that updates the use count and, concurrently therewith, updates the use count flag.

ISSUES

- I. Whether claims 1-10, 24-26 and 34-36 were properly rejected as being unpatentable over Lumetta et al., "Managing Concurrent Access for Shared Memory Active Messages",

IEEE, 1998, pages 272-278 ("Lumetta") in view of Kessler et al. U.S. Patent No. 5,841,973 ("Kessler").

GROUPING OF CLAIMS

Claims 4-7, 25-26 and 35-36 will be argued separately from the remaining claims (1-3, 8-10, 24 and 34).

ARGUMENT

Claims 1-3, 8-10, 24 and 34

This group of claims on appeal contains three independent claims: claims 1, 24 and 34. Claim 1 is representative of the claims on appeal and reads as follows:

1. A method of performing a recoverable operation on a message queue in response to a request by a caller in an information handling system, said method comprising the steps of:
 - storing a use count for said message queue indicating a count of tasks accessing said message queue;
 - storing a use count flag for said caller indicating whether said caller has acquired a lock on said message queue;
 - updating said use count; and
 - atomically with updating said use count, updating said use count flag to indicate whether said caller has acquired a lock on said message queue.

Claims 24 and 34 are similar to claim 1, but are directed to apparatus and a program storage device, respectively.

The claims on appeal stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Lumetta et al., "Managing Concurrent Access for Shared Memory Active Messages", IEEE,

1998, pages 272-278 ("Lumetta"), in view of Kessler et al. U.S. Patent 5,841,973 ("Kessler") (paper no. 12, page 2, ¶ 4). For the reasons stated below, this rejection is untenable and should be reversed by the Board.

In applicants' claimed system, as defined for example in claim 1, a use count is stored for the message queue indicating a count of tasks accessing the queue. A use count flag for the caller is stored indicating whether the caller has acquired a lock on a queue. Atomically with updating the use count, the use count flag is updated to indicate whether the caller has acquired a lock on the queue. This use count may be used for various purposes, such as use count-based responsibility passing where any number of tasks can read various message queue chains, concurrent with queue updates being made (page 5, lines 1-3).

Lumetta discloses various systems for managing concurrent access to concurrent messages. Section 3.1 (pages 274-275) discusses various locking algorithms, including the ticket lock and the Anderson lock (page 275). To obtain a lock in the ticket lock algorithm, a process obtains a ticket and waits for a service counter to show its ticket number. The ticket counter is incremented atomically to ensure that each process receives a different ticket. When a process releases a lock, it increments the service counter to allow the next process waiting on the lock to proceed. The Anderson lock, which is said to improve on the ticket lock, replaces the service counter with a separate flag for each process waiting on the lock. The lock operation obtains a slot assignment with a compare and swap (CAS), then waits for the assigned slot to contain a lock indicator. When releasing a lock, a process moves the lock indicator from its slot into the next.

Kessler describes a messaging facility for a multiprocessor computer system. As described at column 8, lines 24-36, a 64-bit message queue control word (MQCW) 70 (Fig. 5) contains a 21-bit limit field 72 indicating the number of slots 64 (Fig. 4) corresponding to individual messages in a message queue 60. In one of its uses, the value in the limit field 72 is compared with the value in the adjacent tail field 74 (indicating the location of the tail of the queue) to determine whether the message queue is full and cannot accept another message (col. 11, line 65 to col. 12, line 3).

The Examiner argues essentially that Lumetta discloses applicants' claimed combination except for the element of indicating the number of tasks accessing the queue, that Kessler's limit field 72 corresponds to this latter element, and that it would have been obvious in view of Kessler to include this element in Lumetta "because it would have provided an efficient mechanism for serializing operations on the message queue" (paper no. 12, page 3, ¶ 5). Applicants respectfully disagree.

In the first place, it is not true that Lumetta discloses applicants' claimed invention except for the element of indicating the number of tasks accessing the queue. The Examiner equates Lumetta's ticket counter with applicants' use count and Lumetta's lock indicator with applicants' use count flag (paper no. 12, pages 2-3, ¶ 5). However, not only does the ticket counter not indicate the number of tasks accessing the queue (as the Examiner apparently concedes), but it also does not store any other relevant count. Rather, it is merely a means for ensuring that each process receives a different ticket (as in a deli, for example). Thus there is no indication that the ticket counter is ever decremented (as for example in claim 3), as one would expect if it truly represented a use count.

Moreover, Lumetta's ticket counter and lock indicator appear in different implementations as alternatives to each other, and not in a single implementation as interoperating elements as would be typical of an anticipatory reference. Thus, the ticket counter appears in Lumetta's ticket lock, while the lock indicator appears in the Anderson lock as a replacement for the ticket counter (page 275). Since the ticket counter and lock indicator appear in different implementations, they are obviously not updated together atomically, as the use count and use count flag are in applicants' claimed system.

Not only does Lumetta thus lack features attributed to it by the Examiner, but the secondary reference of Kessler does as well. In particular, the limit field 72 is not a "use count" for a message queue indicating a count of tasks accessing the queue, as asserted by the Examiner. While the value of the limit field 72 can be changed, it merely specifies the maximum size of the queue 60 in terms of the number of messages and is relatively static in normal operation. The

limit field 72 does not indicate the actual size of the queue 60—that is what the tail field 74 does—nor does either field 72 or 74 indicate a count of tasks accessing the queue.

Thus, neither reference teaches what it is claimed of it by the Examiner. Accordingly, even if the Kessler's teachings were combined with those of Lumetta in the manner suggested by the Examiner, the resulting combination would not duplicate applicants' claimed invention.

Claims 4-7, 25-26 and 35-36

Claim 4, dependent on claim 1, reads as follows:

4. The method of claim 1, comprising the further step of:
comparing said use count with a previously read use count atomically with said updating steps, said updating steps being performed only if said use count matches said previously read use count.

Claims 5-7 depend on claim 4. Claims 25 and 35 are similar except that they depend on claims 24 and 34, respectively. Claims 26 and 36 depend on claims 25 and 35, respectively.

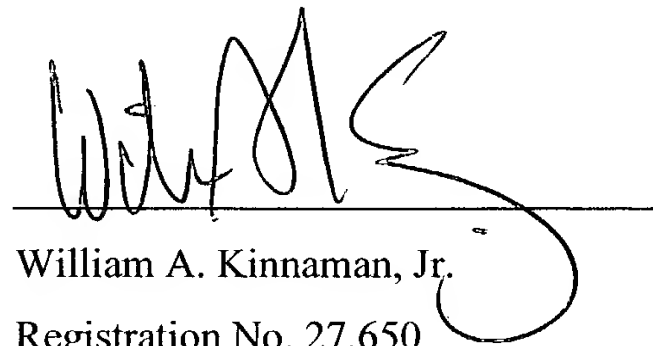
Claims 4, 25 and 35 are believed to distinguish patentably over the art cited by virtue of this additional limitation, in addition to the reasons urged above with respect to claim 1. By atomically comparing the current use count with a previously read use count and conditioning the updating operation on a successful comparison, applicants guard against the possibility that another caller has incremented the use count in the meantime. While Lumetta describes compare-and-swap (CAS) operations, he performs such operations on such entities as a queue tail and packet type (page 275) rather than on a use count as claimed by applicants.

Conclusion

For the foregoing reasons, claims 1-10, 24-26 and 34-36 as amended are believed to distinguish patentably over the art cited by the Examiner. The Examiner's rejection of these claims is therefore untenable and should be reversed.

Respectfully submitted,
DONALD F. AULT et al.

By

A handwritten signature in black ink, appearing to read 'WAK', is written over a horizontal line. The signature is stylized with a large, sweeping flourish extending to the right.

William A. Kinnaman, Jr.

Registration No. 27,650

Phone: (845) 433-1175

Fax: (845) 432-9601

WAK/wak

APPENDIX
Claims on Appeal

1. (Previously presented) A method of performing a recoverable operation on a message queue in response to a request by a caller in an information handling system, said method comprising the steps of:

storing a use count for said message queue indicating a count of tasks accessing said message queue;

storing a use count flag for said caller indicating whether said caller has acquired a lock on said message queue;

updating said use count; and

atomically with updating said use count, updating said use count flag to indicate whether said caller has acquired a lock on said message queue.

2. (Previously presented) The method of claim 1 in which said recoverable operation is a locking operation, said step of updating said use count comprising the step of incrementing said use count, said step of updating said use count flag comprising the step of updating said use count flag to indicate that said caller has acquired a lock on said message queue.

3. (Previously presented) The method of claim 1 in which said recoverable operation is an unlocking operation, said step of updating said use count comprising the step of decrementing said use count, said step of updating said use count flag comprising the step of updating said use count flag to indicate that said caller has released a lock on said message queue.

4. (Original) The method of claim 1, comprising the further step of:

comparing said use count with a previously read use count atomically with said updating steps, said updating steps being performed only if said use count matches said previously read use count.

5. (Original) The method of claim 4 in which said use count is stored in a message queue table having an entry for said message queue.

6. (Previously presented) The method of claim 5 in which said message queue table also stores a pointer to said message queue, said method comprising the further step of:
comparing said pointer with a previously read pointer atomically with said updating steps, said updating steps being performed only if said pointer matches said previously read pointer.
7. (Previously presented) The method of claim 5 in which said message queue table also stores an identifier of said message queue.
8. (Original) The method of claim 1 in which said use count flag is stored in a control block for said caller.
9. (Previously presented) The method of claim 8 in which said control block for said caller also contains an identifier of said message queue.
10. (Original) The method of claim 1 in which said updating steps are performed by executing a single atomic instruction that updates said use count and, concurrently therewith, updates said use count flag.
24. (Previously presented) Apparatus for performing a recoverable operation on a message queue in response to a request by a caller in an information handling system, comprising:
means for storing a use count for said message queue indicating a count of tasks accessing said message queue;
means for storing a use count flag for said caller indicating whether said caller has acquired a lock on said message queue;
means for updating said use count; and
means for updating said use count flag atomically with updating said use count to indicate whether said caller has acquired a lock on said message queue.
25. (Original) The apparatus of claim 24, further comprising:

means for comparing said use count with a previously read use count atomically with said updating operations, said updating operations being performed only if said use count matches said previously read use count.

26. (Previously presented) The method of claim 25 in which said use count is stored in a message queue table having an entry for said message queue, said message queue table also storing a pointer to said message queue, said method comprising the further step of:

comparing said pointer with a previously read pointer atomically with said updating operations, said updating operations being performed only if said pointer matches said previously read pointer.

34. (Previously presented) A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for performing a recoverable operation on a message queue in response to a request by a caller in an information handling system, said method steps comprising:

storing a use count for said message queue indicating a count of tasks accessing said message queue;

storing a use count flag for said caller indicating whether said caller has acquired a lock on said message queue;

updating said use count; and

atomically with updating said use count, updating said use count flag to indicate whether said caller has acquired a lock on said message queue.

35. (Original) The program storage device of claim 34, said method steps further comprising: comparing said use count with a previously read use count atomically with said updating steps, said updating steps being performed only if said use count matches said previously read use count.

36. (Previously presented) The program storage device of claim 35 in which said use count is stored in a message queue table having an entry for said message queue, said message queue table also storing a pointer to said message queue, said method steps further comprising:

comparing said pointer with a previously read pointer atomically with said updating steps, said updating steps being performed only if said pointer matches said previously read pointer.